The Inclusion Criterion for Data Clustering

A Thesis

submitted to the designated
by the General Assembly
of the Department of Computer Science and Engineering
Examination Committee

by

Nikolaos Kornelakis

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN DATA AND COMPUTER SYSTEMS ENGINEERING

WITH SPECIALIZATION
IN DATA SCIENCE AND ENGINEERING

University of Ioannina

January 2021

Examining Committee:

- **Aristidis Likas**, Professor, Department of Computer Science and Engineering, University of Ioannina (Advisor)
- **Konstantinos Blekas**, Assoc. Professor, Department of Computer Science and Engineering, University of Ioannina
- **Konstantinos Vlachos**, Assist. Professor, Department of Computer Science and Engineering, University of Ioannina

Dedication

To my family.

Acknowledgements

I would really like to thank my supervisor Prof. Aristidis Likas for his valuable guidance and support from the first moment until the end of this work.

Table of Contents

List of Figures iii

List of Tables iv

List of Algorithms vi

Abstract vii

Εκτεταμένη Περίληψη ix

1 Introduction 1

1.1 Objectives 2

1.2 Structure 2

2 Preliminaries 3

2.1 Data clustering 3

2.2 Clustering methods 5

2.2.1 K-means 6

2.2.2 Spectral clustering 7

2.3 Clustering quality measures 8

2.3.1 Silhouette criterion 8

2.3.2 Modularity criterion 10

3 Inclusion for binary graphs 13

3.1 Binary inclusion criterion 13

3.2 Inclusion Maximization 15

4 The inclusion criterion for weighted similarity matrices 18

4.1 The generalized inclusion 18

4.2 Adjusting the importance of the two terms 20
4.3 Pointwise inclusion 20
4.4 Greedy Inclusion Optimization 21
4.5 Nearest Inclusion 22

5 Experiments **24**
5.1 Binary graphs 25
5.2 Weighted graphs (binary inclusion) 27
5.3 Weighted graphs 29
5.4 Synthetic 2D datasets 32
5.5 Pointwise inclusion 36
5.6 Inclusion on real datasets 40

6 Conclusions and Future work **44**

Bibliography **46**

List of Figures

2.1 Different ways of clustering the same set of points. 4
2.2 Hierarchical clustering example. 4
2.3 Spectral clustering versus k-means. 8
2.4 Silhouette graphical display using Python's Yellowbrick library. 10

3.1 Binary inclusion example a) 3-clusters solution with $I = 0.85$, b) 4-clusters solution with $I = 0.89$ and c) 5-clusters solution with $I = 0.80$ [1]. 15

4.1 Histogram of object level inclusion. 21

5.1 Ring graph example. 30
5.2 The 3Wings dataset and its ground truth. 33
5.3 The 5Gaussians dataset and its ground truth. 34
5.4 The 7Clusters dataset and its ground truth. 35
5.5 Spectral clustering solutions on 7Clusters dataset. 36
5.6 The 4Gaussians dataset and its ground truth. 36
5.7 Spectral clustering results on 4Gaussians dataset. 38
5.8 Pointwise inclusion histogram for 4Gaussians dataset. 39
5.9 Pointwise inclusion histogram for 5Gaussians dataset. 40

List of Tables

5.1 GNM (binary inclusion) vs Louvain on unweighted synthetic graphs consisting of 500 vertices and 10 clusters. 26
5.2 Binary inclusion vs Modularity using spectral clustering on unweighted synthetic graphs consisting of 500 vertices and 10 clusters. 26
5.3 GNM (binary inclusion) vs Louvain on email-Eu-core network. 27
5.4 GNM with kNN-based approach on weighted graphs. 28
5.5 Louvain with modularity on weighted graphs. 28
5.6 GNM with threshold-based approach on weighted graphs. 28
5.7 Louvain with modularity on weighted graphs. 28
5.8 GNM (inclusion) vs Louvain on weighted synthetic graphs consisting of 500 vertices and 10 clusters. 29
5.9 Ring graph categories. 30
5.10 Ring graph variations. 30
5.11 Ring graphs with 30 vertices and 10 clusters. 31
5.12 Ring graphs with 60 vertices and 15 clusters. 31
5.13 Ring graphs with 150 vertices and 30 clusters. 31
5.14 Ring graphs with 300 vertices and 50 clusters. 32
5.15 Ring graphs with 500 vertices and 10 clusters. 32
5.16 Spectral clustering on 3Wings dataset. 33
5.17 Spectral clustering on 5Gaussians dataset. 34
5.18 Spectral clustering on 7Clusters dataset. 35
5.19 Spectral clustering on 4Gaussians dataset. 37
5.20 Spectral clustering on 5Gaussians dataset. 39
5.21 Spectral clustering on Pendigits subset {2, 3}. 40
5.22 Spectral clustering on Pendigits subset {2, 3, 6}. 41
5.23 Spectral clustering on Pendigits subset {0, 2, 3, 6}. 41

5.24 Spectral clustering on Pendigits subset {0, 2, 4, 6, 8}. 41
5.25 Spectral clustering on Optdigits subset {2, 3}. 41
5.26 Spectral clustering on Optdigits subset {2, 3, 6}. 42
5.27 Spectral clustering on Optdigits subset {0, 2, 3, 6}. 42
5.28 Spectral clustering on Optdigits subset {0, 2, 4, 6, 8}. 42
5.29 Spectral clustering on Iris dataset. 43

List of Algorithms

2.1 K-means algorithm. 6
2.2 Spectral clustering algorithm. 7
3.1 Greedy Node Movement Algorithm [1]. 16

Abstract

Nikolaos Kornelakis, M.Sc. in Data and Computer Systems Engineering, Department of Computer Science and Engineering, School of Engineering, University of Ioannina, Greece, January 2021.
The Inclusion Criterion for Data Clustering.
Advisor: Aristidis Likas, Professor.

Evaluating the quality of clustering solutions is a significant task in data analysis, since it allows the comparison among partitions with different number of clusters. In this way the difficult problem of estimating the true number of clusters can be tackled.

A clustering quality measure called 'inclusion' has been proposed in the context of community detection, ie. partitioning the nodes of graph into clusters (communities), with many edges joining nodes of the same cluster and comparatively few edges joining nodes of different clusters. The inclusion quality measure evaluates how well each node is 'included' in its community by considering both its existing and its non-existing edges. However, it is restricted to the case of binary graphs, where each edge is either zero or one.

We introduce an extension of the inclusion definition for the case of general edge matrices where the weight of each edge could take any value between zero and one. Such a wide extension allows the proposed inclusion criterion to be used for any clustering problem given the similarity matrix between data objects. Two methods are considered to exploit the proposed criterion. The first is a greedy approach that starts with every data object in a separate cluster and tries to improve the inclusion of the partitioning by moving each time a single data object to another cluster. The second method relies on using spectral clustering to obtain solutions with different number of clusters and keeping the best solution according to the inclusion criterion. An alternative inclusion definition (called 'nearest inclusion') is also proposed that

computes inter-cluster similarity as the similarity to the nearest cluster only, instead of considering the similarity to all other clusters.

Experiments have been conducted using synthetic and real world datasets to compare inclusion with well-known criteria of clustering quality, such as modularity and silhouette.

Εκτεταμενη Περιληψη

Νικόλαος Κορνελάκης, Δ.Μ.Σ. στη Μηχανική Δεδομένων και Υπολογιστικών Συστημάτων, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πολυτεχνική Σχολή, Πανεπιστήμιο Ιωαννίνων, Ιανουάριος 2021.
Το Κριτήριο Inclusion για Ομαδοποίηση Δεδομένων.
Επιβλέπων: Αριστείδης Λύκας, Καθηγητής.

Η ομαδοποίηση (clustering) αποτελεί ένα από τα σημαντικότερα προβλήματα της ανάλυσης δεδομένων. Στόχος της ομαδοποίησης είναι η διαμέριση των δεδομένων σε ομάδες (clusters), έτσι ώστε να υπάρχει μεγάλη ομοιότητα μεταξύ των δεδομένων της ίδιας ομάδας και όσο το δυνατόν μικρότερη ομοιότητα με δεδομένα διαφορετικών ομάδων.

Δύο από τις πιο διαδεδομένες τεχνικές ομαδοποίησης είναι οι αλγόριθμοι k-means και spectral clustering. Και οι δύο όμως, απαιτούν ως είσοδο τον αριθμό των ομάδων, ο οποίος στις περισσότερες περιπτώσεις δεν είναι γνωστός. Προκύπτει λοιπόν η ανάγκη μετρικών/κριτηρίων έτσι ώστε να μπορεί αξιολογηθεί η ποιότητα της διαμέρισης που παράγεται από μία μέθοδο ομαδοποίησης.

Τέτοια γνωστά κριτήρια αξιολόγησης της ποιότητας ομαδοποίησης είναι το silhouette και το modularity. Πρόσφατα έχει προταθεί το κριτήριο inclusion το οποίο παρουσιάζει καλές επιδόσεις. Όμως έχει οριστεί μόνο για δυαδικούς πίνακες ομοιότητας με δυαδικές τιμές (δηλαδή η ομοιότητα (similarity) μεταξύ των δεδομένων είναι ένα ή μηδέν), γεγονός που περιορίζει σημαντικά το πεδίο εφαρμογής του.

Βασικός στόχος αυτής της εργασίας είναι η γενίκευση του κριτηρίου inclusion ώστε να μπορεί να χρησιμοποιηθεί σε προβλήματα όπου οι τιμές του πίνακα ομοιότητας λαμβάνουν τιμές στο διάστημα μεταξύ μηδέν και ένα. Προτείνονται διάφορες προσεγγίσεις για την επίτευξη του στόχου αυτού. Οι δύο πρώτες μέθοδοι βασίζονται στο μετασχηματισμό του πίνακα ομοιότητας σε δυαδικό χρησιμοποιώντας

είτε κατωφλίωση είτε λαμβάνοντας υπόψη του κοντινότερους γείτονες κάθε δεδομένου. Στη συνέχεια αξιοποιείται το κριτήριο inclusion όπως έχει ήδη διατυπωθεί για δυαδικούς πίνακες ομοιότητας.

Η σημαντικότερη ωστόσο συνεισφορά της εργασίας είναι μια νέα διατύπωση για το κριτήριο inclusion ώστε να μπορεί να χρησιμοποιηθεί για το γενικό πρόβλημα ομαδοποίησης, όπου η ομοιότητα μπορεί να λάβει οποιαδήποτε τιμή μεταξύ μηδέν και ένα. Ταυτόχρονα παρουσιάζεται μια άπληστη (greedy) μεθοδολογία για τη μεγιστοποίηση του προτεινόμενου κριτηρίου. Επιπλέον προτείνεται και μια τροποποίηση του κριτηρίου (που την ονομάζουμε nearest inclusion) η οποία λαμβάνει υπόψη μόνο την κοντινότερη ομάδα κάθε δεδομένου (εκτός της ομάδας στην οποία ήδη ανήκει) και όχι όλες τις υπόλοιπες ομάδες.

Για την αξιολόγηση των κριτηρίων πραγματοποιήθηκαν πειράματα χρησιμοποιώντας τόσο τεχνητά όσο και πραγματικά σύνολα δεδομένων. Από τη σύγκριση με τα αποτελέσματα που προκύπτουν χρησιμοποιώντας τα γνωστά κριτήρια silhouette και modularity μπορούμε να συμπεράνουμε ότι οι προτεινόμενες μεθοδολογίες στις περισσότερες περιπτώσεις παρέχουν ομαδοποιήσεις συγκρίσιμης ή ανώτερης ποιότητας.

Chapter 1

Introduction

1.1 Objectives

1.2 Structure

Clustering is an important problem with several applications. There are many methods proposed for clustering, but most of them like K-Means and Spectral clustering require the number of clusters as input. Unfortunately, in most of the real problems the number of clusters is unknown. For these problems, to use the previously mentioned methods, one must have a way of evaluating the solutions obtained for different number of clusters and select the best among them.

Quality measures make use of different definitions and aspects of clustering problem, in order to evaluate a clustering solution. Examples of well-known quality measures are silhouette and modularity. This thesis focuses on inclusion, another recently proposed quality measure. Inclusion has desirable properties with most import being that it considers both intra and inter cluster density. Also, experimental results has been shown that inclusion criterion is very effective. However, it is defined only for binary similarity matrices which is very restrictive in real problems.

So, the main goal is the generalization of inclusion quality measure for weighted similarity matrices. To accomplish that, we tested three different approaches and we present the results of the experiments we conducted compared to silhouette and modularity on various datasets.

1.1 Objectives

The objectives of this thesis are the following:

- Study and present data clustering and known methods that are used.
- Study and evaluate inclusion quality measure for clustering using binary similarity matrices.
- Generalize the inclusion criterion for the case of weighted similarity matrices.
- Evaluate the quality of generalized inclusion against other widely used criteria for evaluating clustering solutions.

1.2 Structure

This thesis consists of 6 chapters. In chapter 2 we present, the definition of data clustering problem and some basic categories of clustering methods. Also, we describe in more details K-Means and Spectral clustering, two of the most popular methods for data clustering and two clustering quality measures, silhouette and modularity.

In chapter 3 we focus on another, recently introduced quality measure, called inclusion that can be used for binary similarity matrices. We also present the greedy method proposed for inclusion maximization. In chapter 4 we define the weighted inclusion criterion for general similarity matrices, which is the main contribution of this thesis, along with some variations of this new criterion.

Chapter 5 contains the results of the experiments we conducted, both on synthetic and real-world data to test the quality of the generalized inclusion criterion. Finally, in chapter 6, we summarize the results obtained from the experiments, we present our conclusions and discuss about future work.

Chapter 2

Preliminaries

2.1 Data clustering

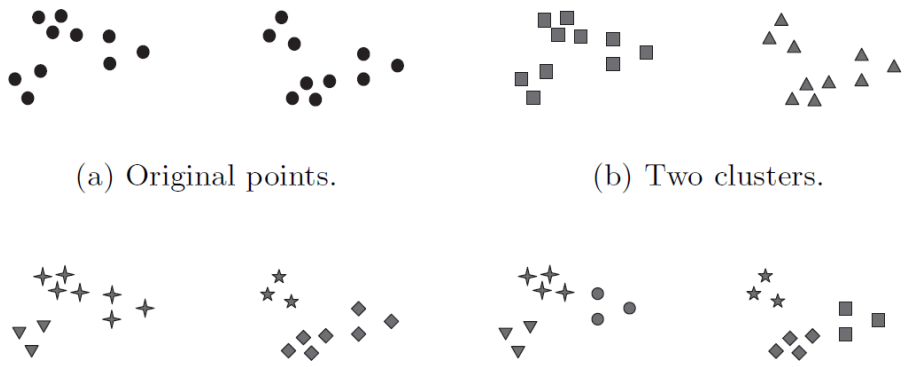
2.2 Clustering methods

2.3 Clustering quality measures

2.1 Data clustering

Data mining is a process of extracting meaningful and useful information from sets of data. Clustering is one of the data mining problems and belongs to unsupervised learning since data are not labeled. It aims to group a set of data objects, so that objects of the same group are similar to each other and different from objects of the other groups.

Part of the difficulty of clustering lies in the fact that in some cases, clusters are not well defined. A characteristic example presented in [2] is shown in figure 2.1 where there are three different reasonable ways of clustering the original points, in two, four and six clusters.

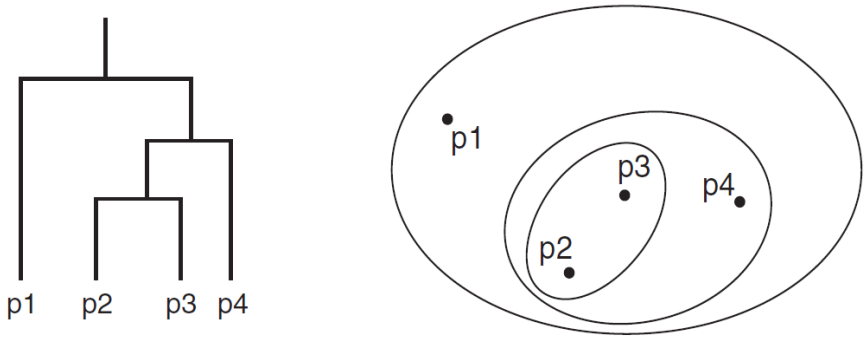


(a) Original points. (b) Two clusters.

(c) Four clusters. (d) Six clusters.

Figure 2.1: Different ways of clustering the same set of points.

Clustering has applications in many practical problems of different areas like social sciences, biology, statistics, information retrieval and more. There are two basic types of clustering: partitional and hierarchical. The main difference between them, is that in partitional clustering the data objects are divided in groups and each data object belongs to exactly one group while in hierarchical clustering groups are overlapping subsets of data objects, forming hierarchies and can be organized as a tree, known as dendrogram (figure 2.2, again from [2]).



(a) Dendrogram. (b) Nested cluster diagram.

Figure 2.2: Hierarchical clustering example.

The most famous method of partitional clustering is k-means, which will be presented in more details in the next section. On the other hand, methods of hierarchical clustering are divided in two categories: agglomerative and divisive. In agglomerative

hierarchical clustering, methods start by setting each data object in a separate group and continue by merging groups until only one group is left, while in divisive hierarchical clustering, methods work with the opposite logic. They start by placing all data object to a single group and continue by dividing groups until each data object belongs to a separate group.

Usually, clustering methods work on the given data objects. There are cases though, in which data objects cannot be provided but only the similarities or the dissimilarities between them. One of these cases is graph clustering or similarity-based clustering. As the name suggests, these methods require the similarity matrix of the data. The most popular of these methods is spectral clustering which will be presented in more detail at the following section.

Graph Partitioning

Graph partitioning is the problem of partitioning the nodes of a graph into k almost equal-sized groups and at the same time minimize the number of edges between different groups [3]. Graph partitioning has many applications in different areas of computer science like VLSI circuit design [4], parallel computing [5] and many more.

It is an NP-hard problem and thus only approximate methods can be used. Despite the difficulty of the problem, there are many heuristic methods proposed, showing good results. Two of the most famous methods are Kernighan–Lin algorithm [6] and Fiduccia–Mattheyses algorithm [7].

Graph partitioning problem is one of the mathematical formalizations of community detection [8] which is a graph clustering problem so it can also be approached with similarity-based clustering methods.

2.2 Clustering methods

In this section we will describe in more detail two of the most popular methods of clustering: k-means and spectral clustering.

2.2.1 K-means

K-means [9] is one of the most popular clustering methods since it is very fast and simple, but effective at the same time. It requires the set of objects in form of d-dimensional vectors $X = \{x_1, x_2, ..., x_n\}$ and the number of clusters k as input.

K-means starts with an initialization step, by choosing randomly k representatives for each cluster usually from the input dataset X, which are called centroids. Next, two steps are iteratively applied. In the first step, k clusters are formed, by assigning each object to the cluster of the closest centroid. At the second step, the centroid of each cluster is recomputed as the mean of all the objects assigned to its cluster. This process continues until centroids stop getting updated.

Algorithm 2.1 K-means algorithm.

Require: input data as d-dimensional vectors and the number of clusters k

1: Select randomly k objects as the initial centroids.
2: **repeat**
3: Form k clusters by assigning each object to the cluster of the closest centroid.
4: Recompute the centroids for each cluster.
5: **until** centroids stop getting updated

K-means aims to minimize the sum squared error of the distances between each object x of each cluster C_i and the centroid c_i of that cluster, formulated as follows:

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - c_i\|^2 \tag{2.1}$$

The centroid c_i in the above formula is calculated as:

$$c_i = \frac{1}{n_i} \sum_{x \in C_i} x \tag{2.2}$$

where n_i is the number of objects in cluster C_i.

Although k-means has linear complexity and good results in general, it has some disadvantages too. One of them is the high dependence on the initial positions of centroids. In order to avoid this, one can run k-means multiple times and keep the solution with the minimum error. Another disadvantage is that it is affected from outliers. They can shift the centroids away from the center of the clusters and distort

the result. This can be avoided by removing outliers from data as a preprocessing step. Finally, k-means has troubles when clusters of the data have different size and density. It tends to create compact circular groups and break big clusters.

2.2.2 Spectral clustering

Spectral clustering is a technique introduced by Ng, Jordan and Weiss in [10]. Given the similarity matrix, consisting of the pairwise similarities of the data objects, and the number of clusters, spectral clustering performs dimensionality reduction using the spectrum of similarity matrix and then applies a standard clustering method, usually k-means, to the transformed data.

Algorithm 2.2 Spectral clustering algorithm.

Require: $m \times m$ similarity matrix S of the data and the number of clusters k

1: Define D as the diagonal matrix whose (i, i)-element is the sum of S's i-th row
2: Compute the Laplacian $L = D^{-1/2} S D^{-1/2}$
3: Compute the k largest eigenvectors of L
4: Create an $m \times k$ matrix Y with the k eigenvectors in its columns
5: Run k-means with k clusters and Y matrix as input

As mentioned above, spectral clustering requires as input the similarity matrix. In cases where data are provided and distances between them can be calculated, one can easily convert distances to similarities using many different ways given the data vectors x and x', like:

- **RBF Similarity**:

$$S(x, x') = e^{\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)} \tag{2.3}$$

- **Cosine Similarity**:

$$S(x, x') = \frac{x x'^T}{\|x\| \, \|x'\|} \tag{2.4}$$

Spectral clustering has many advantages compared to other clustering algorithms with the most important being that it is not affected from clusters having different size or shape. The following figure 2.3 shows one of the most typical examples, comparing spectral clustering and k-means on a synthetic dataset that consists of 900 objects divided in 3 clusters.

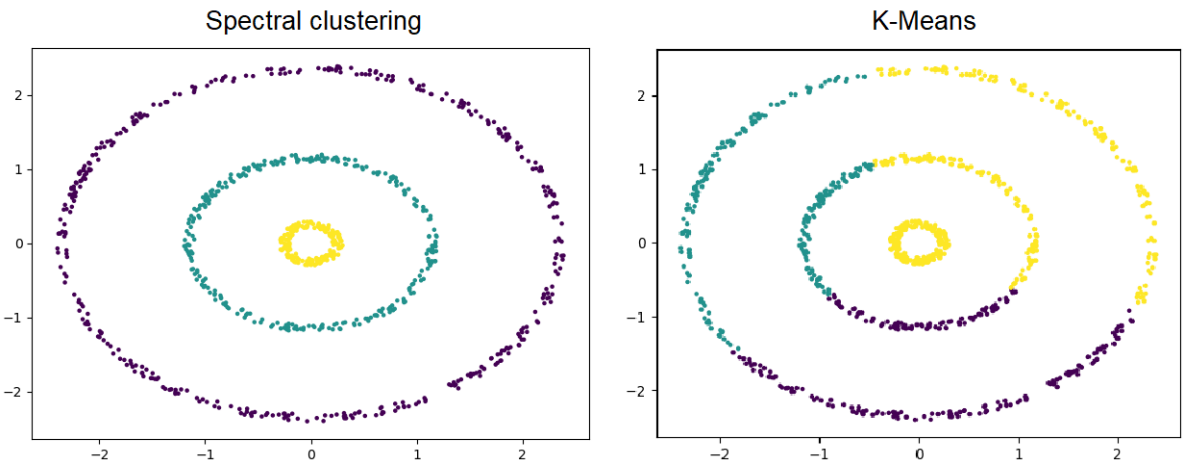


Figure 2.3: Spectral clustering versus k-means.

The main disadvantage of spectral clustering is that it cannot be used in large datasets since it has computational complexity $O(n^3)$.

2.3 Clustering quality measures

Quality measures are used to evaluate a clustering solution of a set of objects. In most of the real-world clustering problems the number of clusters is unknown, meaning that spectral clustering, k-Means, or other clustering techniques that require the number of clusters as input cannot be used directly.

Clustering quality measures help dealing with this problem. The selected clustering technique can be applied multiple times for different number of clusters k and the obtained solutions can then be evaluated by a quality measure. In the end, the solution of highest quality will be chosen as the final solution of the clustering problem.

2.3.1 Silhouette criterion

Silhouette is a criterion introduced by Rousseeuw in [11]. It is used not only for the evaluation of a clustering result, but also for its interpretation since silhouette provides a graphical display, showing for each object how well it fits in its cluster and how well it is separated from objects of other clusters.

Silhouette starts by calculating for each object i the average dissimilarity/distance

to all other objects that belong to the same cluster with i, using the following formula:

$$a\left(i\right)=\frac{1}{\left|C_i\right|-1}\sum_{j\in C_i,\ i\neq j}d(i,j) \tag{2.5}$$

where $d\left(i,j\right)$ is the dissimilarity between objects i and j and $|C_i|$ is the number of objects in the cluster C_i. This is used to evaluate the wellness of each object in its cluster.

Then silhouette calculates again for each object i, the average dissimilarity to all objects that belong to the nearest cluster of C_i.

$$b\left(i\right)=min_{k\neq i}\frac{1}{\left|C_k\right|}\sum_{j\in C_k}d(i,j) \tag{2.6}$$

The nearest cluster of C_i is also called neighboring cluster. Using $b\left(i\right)$ silhouette evaluates how well each object is separated from objects out of its cluster.

Combining $a\left(i\right)$ and $b\left(i\right)$, the final evaluation of each object i is obtained from:

$$s\left(i\right)=\begin{cases}1-a\left(i\right)/b\left(i\right), & \text{if } a\left(i\right)<b\left(i\right)\\ 0, & \text{if } a\left(i\right)=b\left(i\right)\\ b\left(i\right)/a\left(i\right)-1, & \text{if } a\left(i\right)>b\left(i\right)\end{cases} \tag{2.7}$$

or using a single formula:

$$s\left(i\right)=\frac{b\left(i\right)-a(i)}{max\{a\left(i\right),b\left(i\right)\}} \tag{2.8}$$

Having $s\left(i\right)$ calculated per object, one can easily calculate the silhouette of the clustering solution by averaging $s(i)$ of all the data objects.

In case the similarities are provided as input, instead of dissimilarities, silhouette can still be calculated with two small modifications. One is at the calculation of $b\left(i\right)$, where we need maximum instead of minimum and the other is at the calculation of $s\left(i\right)$, updated as:

$$s'\left(i\right)=\begin{cases}1-b'\left(i\right)/a'\left(i\right), & \text{if } a'\left(i\right)>b'\left(i\right)\\ 0, & \text{if } a'\left(i\right)=b'\left(i\right)\\ a'\left(i\right)/b'\left(i\right)-1, & \text{if } a'\left(i\right)<b'\left(i\right)\end{cases} \tag{2.9}$$

Silhouette outputs a value in $[-1,1]$. In case where there is only one cluster it cannot be defined so in this case output value is zero. Also, it can be used with any distance, but Euclidean is the most frequently used.

Finally, as mentioned in the beginning, silhouette can also provide a graphical display to ease the evaluation of a clustering solution of a set of objects using the $s(i)$ values of objects.

Figure 2.4 provides a graphical presentation of silhouette, produced using Python's Yellowbrick library, after applying k-means to a dataset with 2000 objects divided in 10 clusters.

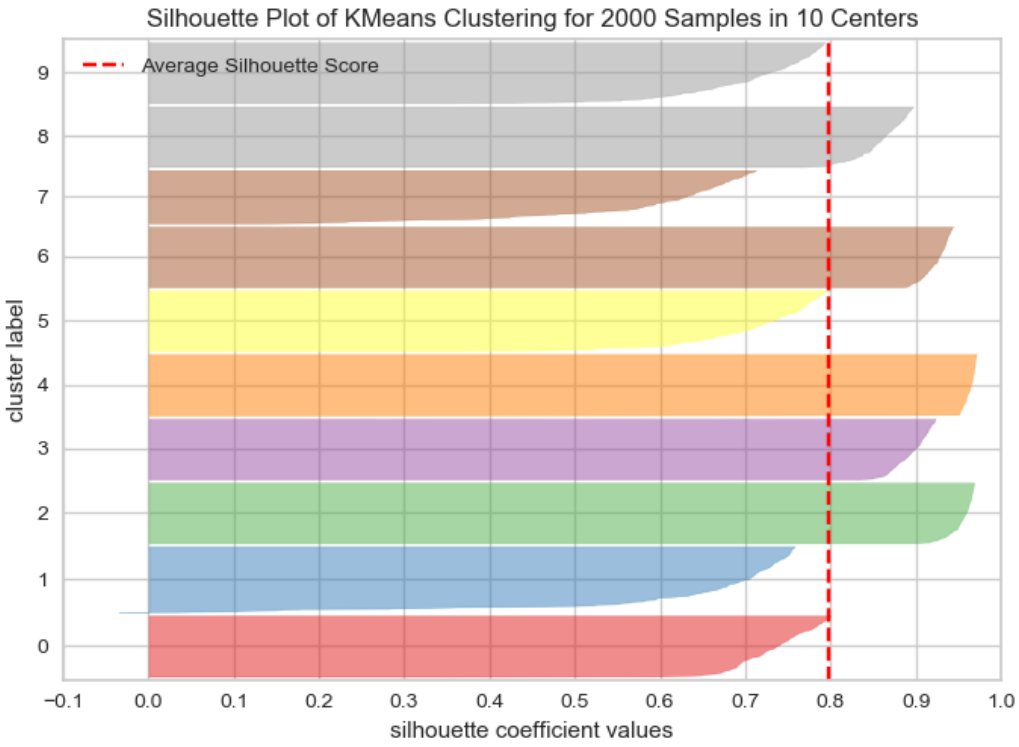


Figure 2.4: Silhouette graphical display using Python's Yellowbrick library.

2.3.2 Modularity criterion

As already mentioned, community detection is equivalent to the graph clustering problem. The term network refers to the graph, nodes refer to the vertices of graph and communities to clusters. If the nodes of a network are naturally divided into groups having high internal edge density and at the same time low edge density between them, then the network is said to have community structure.

Modularity is a known quality measure introduced by Newman and Girvan [12] and is very often used in community detection to evaluate the strength of a community structure. The concept behind modularity is that in a graph having community

structure, communities are expected to have more internal edges than in a random graph with the same number of nodes and same node degrees.

Modularity is defined as follows:

$$Q = \frac{1}{2m} \sum_{i,j} \left[w_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i,\ c_j) \tag{2.10}$$

where w_{ij} is the edge weight between node i and j, $m = \frac{1}{2} \sum_{ij} w_{ij}$ is the sum of weights of the graph, $k_i = \sum_j w_{ij}$ is the sum of weights of the edges attached to node i, c_i is the community in which node i belongs and $\delta(u, v)$ outputs 1 if $u = v$ and 0 otherwise.

One disadvantage of modularity, proved by Fortunato and Barthélemy in [13], is that it suffers from resolution limit and hence, it fails to detect small communities. Also, Brandes, Delling, Gaertler, Gorke, Hoefer, Nikoloski and Wagner proved in [14] that exact modularity optimization is an NP-Complete problem meaning that, at least in large networks, only approximate methods can be used. The most popular method among them is Louvain Algorithm.

Louvain is a greedy algorithm introduced by Blondel, Guillaume, Lambiotte and Lefebvre in [15] and consists of two phases. The first phase starts by initializing each node to a separate community. Next for each node i, the algorithm calculates the gain of modularity by moving node i to its neighbor's communities one by one and finally places the node to the community with the maximum gain. If the maximum gain is negative, then the node stays in its cluster. This process continuous until modularity stop improving.

The second phase starts by creating a new graph having as nodes the communities occurred from the previous phase. The edges between two nodes i and j in the new graph, are calculated as the sum of all the edges between nodes of communities C_i and C_j in the original graph. Internal edges of a community in the original graph are interpreted as self-loops for this node in the new graph. After that, the same iteration of the first phase is applied again in the new graph. These two phases iteratively applied until improvement of modularity stops.

The order of the nodes affects the algorithm's computation time, but in general Louvain algorithm is fast with time complexity $O(nlog_2 n)$ where n is the number of nodes in graph. Another reason for being fast is because modularity gain ΔQ by moving an isolated node to a community C is very easy to calculate with the following

formula:

$$\Delta Q = \left[\frac{\sum_{in} + k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right] \tag{2.11}$$

where $\sum_{in}$ is the sum of weights of internal edges of C, $\sum_{tot}$ is the sum of weights of edges incident to nodes in C, k_i is the sum of weights of edges incident to node i, $k_{i,in}$ is the sum of weights of edges from node i to nodes of community C and m is the sum of weights of all edges in the graph. Another formula, like the above, exists to calculate the gain of removing a node from its community.

Chapter 3

Inclusion for binary graphs

3.1 Binary inclusion criterion

3.2 Inclusion Maximization

In this chapter we present the binary inclusion criterion for community detection. Given the fact that community detection is equivalent to graph clustering, as mentioned in section 2.3.2, and that a similarity matrix can be represented as a graph, inclusion could be presented as a criterion for data clustering when binary similarity matrix is provided (consisting only of 0s and 1s). However, for literature consistency matters, we will present inclusion in terms of community detection.

3.1 Binary inclusion criterion

Inclusion is a quality measure used for community evaluation and detection in undirected and unweighted graphs, introduced by Koufos and Likas [16] [1]. The main idea behind this quality measure, is that a node is well placed in a community if it has many edges with other nodes of the same community and as few edges as possible with nodes of other communities, considering both inter and intra cluster density.

So, the goal is the maximization of two quantities one for inter and one for intra cluster density. The inter cluster density for a node is formulated as the number of edges to nodes of the same community divided by the degree of the node. On the

other hand, the intra cluster density for a node is formulated as the number of non-existing edges with nodes outside its community divided by the number of nodes in the graph minus the degree of the node. Note that graph is unweighted, so edge weights can be either 0 or 1.

The interpretation of the above description is that a node is perfectly placed in a community when it has edges only to nodes of this community, which is pretty much the definition of community detection problem. More specifically, given a graph and community structure $C = \{C_1, C_2, \ldots, C_k\}$, inclusion is calculated per node and is defined as follows:

$$I_i^B = \frac{I_i^1\,(in) + \ I_i^0(out)}{2} = \frac{1}{2}\left(\frac{e_i^1(in)}{d_i} + \frac{e_i^0\,(out) + 1}{n - d_i}\right) \tag{3.1}$$

where n is the number of nodes in the graph, d_i is the degree of each node i, $e_i^1(in)$ is the number of existing edges between node i and nodes in its community and $e_i^0\,(out)$ is the number of none-existing edges between node i and nodes out of its community.

The final inclusion is defined as the average inclusion of all the nodes in the graph:

$$I^B = \frac{1}{n}\sum_{i=1}^{n} I_i \tag{3.2}$$

The maximum value of inclusion ($I_i^B = 1$) occurs when the graph is complete and all the nodes belongs to the same community, or multiple disconnected complete subgraphs and each subgraph forms a different community. The smallest value ($I_i^B = 0.5$) occurs for the previously mentioned graphs, but this time when each node forms its own community.

Inclusion has desirable properties. The most important is that it considers both intra and inter cluster density, since the inclusion of a node is calculated based on how well the node fits in its community ($I_i^1\,(in)$) and how well the node is separated from the nodes out of its community ($I_i^0\,(out)$).

Figure 3.1 shows an example of binary inclusion presented in [1] for 3 different partitions of the same graph. Visually, one can easily understand that the best partitioning for this graph is (b). Partitioning (a) is a good enough while (c) is the worst. Inclusion agrees with this intuition, since it gives the maximum score $I^B = 0.89$ to solution (b), the second maximum score $I^B = 0.85$ to (a) and the worst $I^B = 0.80$ to (c).

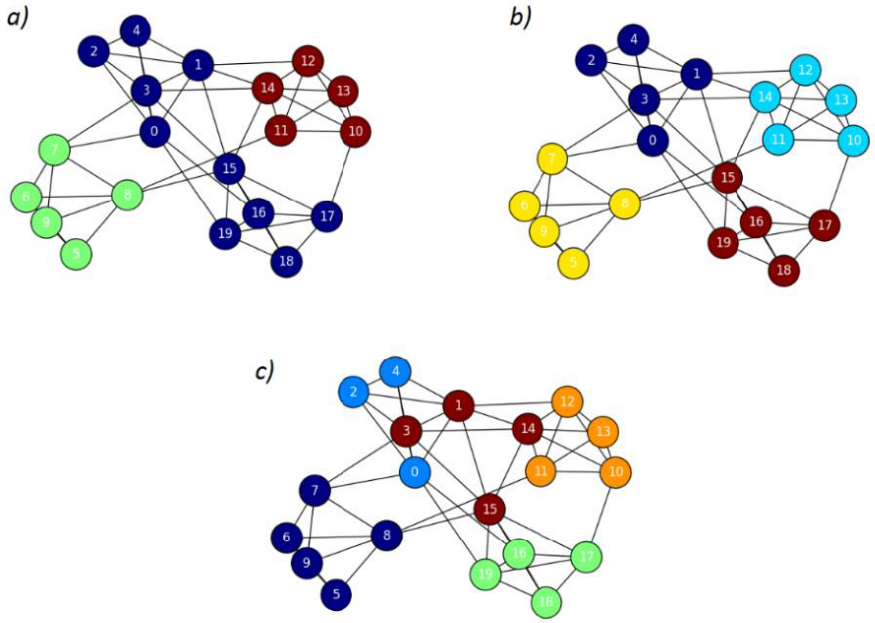


Figure 3.1: Binary inclusion example a) 3-clusters solution with $I = 0.85$, b) 4-clusters solution with $I = 0.89$ and c) 5-clusters solution with $I = 0.80$ [1].

3.2 Inclusion Maximization

In [1] Koufos and Likas also formulated a search strategy, inspired by Louvain algorithm, for finding communities by maximizing inclusion criterion. This approximate method is called Greedy Node Movement since, as its name suggests it is a greedy search algorithm.

Greedy Node Movement starts with an initialization step by setting all the nodes of the graph in separate communities. After that, the algorithm is iteratively moving each time a single node between communities, by calculating for each possible node movement the difference in graph inclusion caused by the movement of the node i from community C_k to community C_l, as follows:

$$\Delta I_{1i}^{B} = \frac{1}{2} \sum_{j \in C_k,\ j \neq i} \left\{ (1 - e_{ij}) \left(\frac{1}{n - d_j} + \frac{1}{n - d_i} \right) - e_{ij} \left(\frac{1}{d_j} + \frac{1}{d_i} \right) \right\} \tag{3.3}$$

$$\Delta I_{2i}^{B} = \frac{1}{2} \sum_{j \in C_l} \left\{ e_{ij} \left(\frac{1}{d_j} + \frac{1}{d_i} \right) - (1 - e_{ij}) \left(\frac{1}{n - d_j} + \frac{1}{n - d_i} \right) \right\} \tag{3.4}$$

where n is the number of nodes in the graph, d_i is the degree of node i and e_{ij} is the

weight of the edge between node i and j. Note here that since graphs are unweighted, the weight between two nodes can be either 0 or 1, so only one of the two terms of ΔI_{1i} and ΔI_{2i} must be calculated each time. In the above formula ΔI_{1i} expresses the gain of removing a node from its community and ΔI_{2i} expresses the gain of moving a node to the new community.

Combining ΔI_{1i} and ΔI_{2i} the difference in graph inclusion by a node movement from community C_k to community C_l, can be calculated as follows:

$$\Delta I_i^B (C_k, C_l) = \frac{1}{n}(\Delta I_{1i} + \Delta I_{2i}) \tag{3.5}$$

The algorithm stops when there is no possible node movement that improves the graph inclusion.

Algorithm 3.1 Greedy Node Movement Algorithm [1].

Require: A graph $G = (V, E)$

Ensure: A partition $C = \{C_1, C_2, \ldots, C_k\}$ of the graph

$C \leftarrow$ Set all nodes to separate communities
repeat
  max$\Delta I^B \leftarrow 0$
  for node $n \in V$ **do**
    for community $c \in C$ **do**
      Calculate ΔI^B
      if $\Delta I^B >$ maxΔI^B **then**
        Store n, c with the current maximum ΔI^B
      end if
    end for
  end for
  Move node n to community c that resulted in maxΔI^B
until graph inclusion is not getting improved

Some alternative strategies have also been tested in [1], in order to improve the algorithm both from computational complexity and accuracy perspective. The first one is to move a node from a community to another without necessarily having the maximum ΔI^B but just positive. This strategy reduces the complexity of algorithm and at the same time ensures that total inclusion is improved at each step. Another

strategy tested was to check only the neighbor community of the node for possible movement. The final variation tested, refers to the examination order of the nodes, sequentially or randomized.

Chapter 4

The inclusion criterion for weighted similarity matrices

4.1 The generalized inclusion

4.2 Adjusting the importance of the two terms

4.3 Pointwise inclusion

4.4 Greedy Inclusion Optimization

4.5 Nearest Inclusion

4.1 The generalized inclusion

As mentioned in the previous chapter, although inclusion seems to have good results, it is defined only for binary similarity matrices, which seems to be restrictive since most of the real-world problems are defined using weighted ones.

In order to apply inclusion in general similarity matrices we examined three different approaches:

- k-nearest neighbor(kNN)-based
- threshold-based
- generalization of inclusion definition

The first two approaches are simple and similar to each other. They use binary inclusion as presented in the previous chapter but before that, they transform the weighted similarity matrix to binary in a preprocessing step. For each object, kNN-based approach assigns 1 to the k maximum and 0 to the rest weights of similarity matrix, where k is a user specified parameter. Similarly, threshold-based approach assigns 1 to the weights greater than a threshold t and 0 to the rest weights of similarity matrix, where t is again a user specified parameter.

The third and most important approach is the generalization of binary inclusion itself. We now define the new inclusion measure for general similarity matrices with weights $w_{ij} \in [0, 1]$. So, given a similarity matrix and a clustering solution $C = \{C_1, C_2, ..., C_k\}$, the inclusion of a data object i is defined as:

$$I_i = \frac{1}{2}\left(\frac{w_{ij}^1}{d_i^1} + \frac{w_{ij}^0 + 1}{d_i^0}\right) \tag{4.1}$$

where:
$w_{ij}^1 = \sum_{j \in C_i} w_{ij}$ is the sum of weight between object i and the rest objects in its cluster,
$w_{ij}^0 = \sum_{j \notin C_i} (1 - w_{ij})$ corresponds to the sum of weights of "non-existing" edges of the binary inclusion, between object i and objects outside its cluster,
$d_i^1 = \sum_j w_{ij}^1 = \sum_j w_{ij}$ is the sum of weights of object i with all the other objects,
$d_i^0 = \sum_j w_{ij}^0 = \sum_j (1 - w_{ij}) = N - d_i^1$ corresponds to the sum of weights of "none-existing" edges of binary inclusion, between object i and the rest data objects.

Thus, inclusion is computed using the following more analytic formula:

$$I_i = \frac{1}{2}\left(\frac{\sum_{j \in C_i} w_{ij}}{\sum_j w_{ij}} + \frac{\left[\sum_{j \notin C_i} (1 - w_{ij})\right] + 1}{N - \sum_j w_{ij}}\right) \tag{4.2}$$

As one can observe in the above formula, we kept an analogy between binary inclusion and the new inclusion for general similarity matrices, in order to keep all the good properties of the original inclusion. Hence, object level inclusion calculation consists again of two parts. The first part expresses how well the object i fits in its cluster and the second part expresses how well object i is separated from objects of other clusters. Therefore both intra and inter cluster density are considered.

The total inclusion of a given clustering solution is calculated again by averaging the object level inclusion:

$$I = \frac{1}{n} \sum_{i=1}^{n} I_i \tag{4.3}$$

In the case of a graph, we can see that in the ideal case where the graph consists of a single complete graph and all nodes belong to the same cluster, or multiple disconnected complete subgraphs and each subgraph forms a different cluster, new inclusion's value is also maximized. In the worst case when each node forms its own cluster, inclusion gets its minimum value.

4.2 Adjusting the importance of the two terms

Until now, object level inclusion considers both intra and inter clusters density equally. A new parameter can be added, in order to give inclusion the capability to put more emphasis on one of them. Parameter $a \in [0, 1]$ is included in the object level inclusion formula as follows:

$$I_i = a \frac{w_{ij}^1}{d_i^1} + (1 - a) \frac{w_{ij}^0 + 1}{d_i^0} \tag{4.4}$$

High values of a make inclusion focus more on intra cluster density, while low values of a on inter cluster density. The total inclusion calculation remains the same.

$$I = \frac{1}{n} \sum_{i=1}^{n} I_i \tag{4.5}$$

4.3 Pointwise inclusion

As we have already mentioned, given a similarity matrix and a clustering solution, inclusion is calculated by averaging the object level inclusion. This additional information can be visualized and used to interpret the final inclusion score.

Suppose that we apply spectral clustering with several different values for parameter k, using inclusion as a quality measure, to find the best clustering solution. For each candidate solution, along with the final inclusion score, we can extract additional information by constructing a histogram of object level inclusion scores. Each bin in the histogram corresponds to a value of k and the value of each bin represents the number of objects that presented their maximum inclusion score for that k. If the maximum inclusion score of an object occurs for more than one k, the object will be

counted in all the corresponding bins of the histogram. An example is presented in figure 4.1.

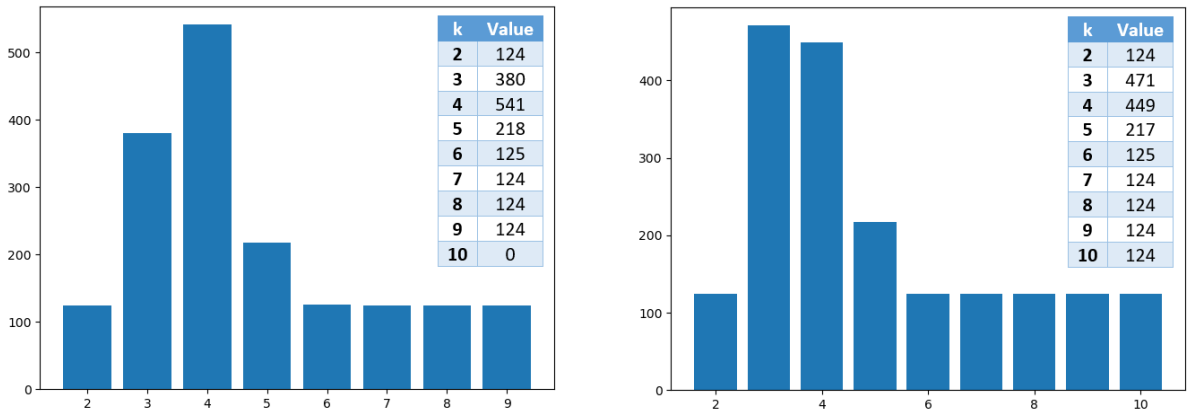


Figure 4.1: Histogram of object level inclusion.

If a bin has much higher value than the other bins, one can be more confident about the solution proposed by inclusion. If multiple bins have values close to the maximum, further investigation may be needed.

In other words, the produced histogram can also be used as an additional criterion to decide which candidate solution is the best.

4.4 Greedy Inclusion Optimization

The Greedy Node Movement algorithm that we described in the previous chapter, can also be used to optimize the new general inclusion criterion. The steps of the algorithm remain the same, but since inclusion for general similarity matrices is calculated differently, the inclusion gain ΔQ by moving an object from cluster C_k to C_l must be updated as follows:

$$\Delta I_{1i} = \frac{1}{2} \sum_{j \in C_k, j \neq i} \left\{ (1 - w_{ij}) \left(\frac{1}{n - \sum_{x \in X} w_{jx}} + \frac{1}{n - \sum_{x \in X} w_{ix}} \right) - w_{ij} \left(\frac{1}{\sum_{x \in X} w_{jx}} + \frac{1}{\sum_{x \in X} w_{ix}} \right) \right\} \tag{4.6}$$

$$\Delta I_{2i} = \frac{1}{2} \sum_{j \in C_l, j \neq i} \left\{ w_{ij} \left(\frac{1}{\sum_{x \in X} w_{jx}} + \frac{1}{\sum_{x \in X} w_{ix}} \right) - (1 - w_{ij}) \left(\frac{1}{n - \sum_{x \in X} w_{jx}} + \frac{1}{n - \sum_{x \in X} w_{ix}} \right) \right\} \tag{4.7}$$

$$\Delta I_i (C_k, C_l) = \frac{1}{n} (\Delta I_{1i} + \Delta I_{2i}) \tag{4.8}$$

where X is the set of objects, w_{ij} is the weight between object i and j in similarity matrix and n is the total number of objects.

One improvement we tested, in order to make Greedy Node Movement faster was to check the gain of inclusion considering only the k nearest clusters of an object instead of all the neighbor clusters, where k is a parameter. This improvement does not seem to compromise the quality of the algorithm's performance and decreases the time complexity of the algorithm.

4.5 Nearest Inclusion

As already mentioned, inclusion considers both intra and inter density of clusters. If an object is placed in a cluster, inter cluster density is increased, as the number of almost zero weights between the object and objects of different clusters increase.

An alternative modeling of inter cluster density is to consider the weights between an object and the objects of its closest cluster only. This can be achieved by first calculating the inclusion of an object i with each one of its outer clusters separately, and then keep the minimum value as the final inclusion. Based on this idea, the nearest inclusion criterion can be formulated as follows:

$$I_i^N = min_{C_m \in C - C_i} \left[\frac{1}{2} \left(\frac{\sum_{j \in C_i} w_{ij}}{\sum_{j \in (C_i \cup C_m)} w_{ij}} + \frac{\left[\sum_{j \in C_m} (1 - w_{ij}) \right] + 1}{N - \sum_{j \in (C_i \cup C_m)} w_{ij}} \right) \right] \tag{4.9}$$

$$I^N = \frac{1}{n} \sum_{i=1}^{n} I_i \tag{4.10}$$

where C_i is the cluster of object i, w_{ij} is the weight between objects i and j and n is the total number of objects.

Note that the formulation of nearest inclusion is based on the way silhouette criterion is defined, which considers only the neighboring cluster for the evaluation of inter cluster density.

The main advantage of inclusion against silhouette criterion is that inclusion is formulated to maximize both parts of intra and inter cluster density, while silhouette is formulated to minimize the part of intra and maximize the part of inter cluster density.

Chapter 5

Experiments

5.1 Binary graphs

5.2 Weighted graphs (binary inclusion)

5.3 Weighted graphs

5.4 Synthetic 2D datasets

5.5 Pointwise inclusion

5.6 Inclusion on real datasets

For the conduction of the experiments we used Python. Nearest inclusion score is denoted as 'N-Inclusion' in the results and it is not presented in experiments that we used greedy node movement since ΔI^N, the gain of nearest inclusion after moving a node from a cluster to another, is not expressed yet. To evaluate the quality of a solution compared to ground truth we used two metrics: Normalize Mutual Information and Adjusted Rand Index.

Mutual Information (MI) in information theory, is a measure of mutual dependence between two random variables [17]. It expresses the 'amount of information' that knowing either variable provides about the other. NMI is the normalized MI to scale its result between $[0, 1]$, with 0 meaning no mutual information and 1 perfect correlation. In data clustering NMI is used to express the shared information between a pair of clustering solutions [18].

Rand Index (RI) is a measure to evaluate the similarity between two clustering solutions by examining all the possible pair combinations and counting those that are

not in the same cluster [19]. RI produces values in $[0, 1]$. Adjusted RI (ARI) corrects RI by ignoring permutations that would have happened by chance, since RI does not ensure that random assignments will produce a value close to 0. The ARI produces values in $[-1, 1]$, with value 1 meaning exact match and value -1 completely different clusterings.

5.1 Binary graphs

As a first step, we implemented the existing binary inclusion and the Greedy Node Movement (GNM) algorithm, proposed for the optimization of this quality measure, in C++. We conducted experiments on the different categories of synthetic unweighted graphs presented in [16] and [1]:

- **B&D**: Balanced and Dense clusters
- **B&DD**: Balanced clusters of Decreasing Density
- **DL&SS**: Dense Large clusters and Sparse small clusters
- **DS&DD**: Decreasing Size and Decreasing Density clusters
- **IS&DD**: Increasing Size and Decreasing Density clusters

Graphs of B&D category consist of equally distributed clusters having high probability in $[0.8, 1]$ for intra-edges. This category results to graphs with clearly separated clusters. B&DD category contains graphs with equally distributed clusters but this time, the probability for intra-edges starts from $[0.9, 1]$ and decreases for each cluster by a constant amount 0.15. DL&SS category contains graphs with high intra-edge probability in $[0.8, 1]$ for each cluster and decreasing cluster size. Graphs of DS&DD category consist of clusters with decreasing both size and intra-edge density. Finally, IS&DD category contains graphs having decreasing intra-edge density and increasing cluster size at the same time. This category results to graphs with large sparse and small dense clusters.

For each category, the results of 10 different executions of the experiment on graphs consisting of 500 vertices and 10 clusters, comparing GNM with inclusion and Louvain algorithm with modularity, are summarized in table 5.1.

Table 5.1: GNM (binary inclusion) vs Louvain on unweighted synthetic graphs consisting of 500 vertices and 10 clusters.

	GNM				Louvain algorithm			
Graph	**Clusters**	**Inclusion**	**NMI**	**ARI**	**Clusters**	**Modularity**	**NMI**	**ARI**
B&D	7.8	0.6338	0.9282	0.7970	7.7	0.1708	0.9238	0.7807
B&DD	8.2	0.6182	0.9390	0.8174	7.4	0.1521	0.9146	0.7644
DL&SS	7.5	0.6703	0.9512	0.9172	6.3	0.2080	0.9100	0.8604
DS&DD	7.1	0.6340	0.9387	0.8627	6.2	0.1700	0.9015	0.8251
IS&DD	8.2	0.6627	0.9558	0.9234	6.2	0.1978	0.8912	0.8333

Next, we conducted the same experiment but this time using spectral clustering with inclusion and modularity as quality measures. The results are presented in the table 5.2.

Table 5.2: Binary inclusion vs Modularity using spectral clustering on unweighted synthetic graphs consisting of 500 vertices and 10 clusters.

	Binary inclusion				Modularity			
Graph	**Clusters**	**Inclusion**	**NMI**	**ARI**	**Clusters**	**Modularity**	**NMI**	**ARI**
B&D	10	0.6403	1	1	10	0.1805	1	1
B&DD	10	0.6235	1	1	10	0.1611	1	1
DL&SS	8.7	0.6731	0.9881	0.9888	8.5	0.2105	0.9859	0.9869
DS&DD	8.5	0.6384	0.9788	0.9814	8.2	0.1756	0.9754	0.9782
IS&DD	8.1	0.6651	0.9733	0.9752	7.6	0.2026	0.9605	0.9476

It can be observed that in both cases (comparing GNM to Louvain algorithm and inclusion to modularity using spectral clustering) the use of inclusion leads to better results for all the different categories of graphs. In some cases the superiority is significant.

Email-Eu-network dataset

The next step was to test the quality of inclusion compared to modularity in a real-world network with known ground truth. Email-Eu-core network[1] is a dataset consisting of 1005 vertices separated in 42 clusters. Each node represents a member of a

[1] https://snap.stanford.edu/data/email-Eu-core.html

research institute and there is an edge between two members if they have exchanged at least one email. Each member belongs to exactly one of the 42 departments of the research institute. This information has been used for the generation of ground truth.

We compared the results of GNM and Louvain algorithm for the above dataset and we present the results in table 5.3.

Table 5.3: GNM (binary inclusion) vs Louvain on email-Eu-core network.

GNM				**Louvain Algorithm**			
Clusters	**Inclusion**	**NMI**	**ARI**	**Clusters**	**Modularity**	**NMI**	**ARI**
135	0.74354	0.7295	0.5455	27	0.4308	0.5957	0.3306

Even though the number of clusters occurred in both cases differs a lot from the original, the solution of GNM with inclusion shows better values for both NMI and ARI measures which indicates and overall better solution.

5.2 Weighted graphs (binary inclusion)

To evaluate kNN-based and threshold-based approaches on weighted graphs, we used a graph of B&D category, consisting of 700 vertices and 20 clusters. In order to add weights to the binary graphs, we used two Normal distributions, one with mean 0.8 and std 0.1 for the existing edges of the graph and another one with mean 0.2 and std 0.1 for the non-existing edges.

KNN-based approach

In this experiment we tried different values for parameter $k \in \{33, 34, 35, 36, 37\}$, since in a graph with 700 vertices and 20 clusters we expect for a vertex to have around 35 important edges. Finally, we kept as solution, the one that yield the maximum score of inclusion. We also applied Louvain algorithm with modularity on the same graph. The results are presented in tables 5.4 and 5.5.

Table 5.4: GNM with kNN-based approach on weighted graphs.

Clusters	Inclusion	NMI	ARI	K
13	0.59968	0.67269	0.42766	33
13	0.59537	0.59806	0.37363	34
11	0.60019	0.66221	0.44468	35
15	0.60162	0.83517	0.63133	36
14	**0.60397**	0.84557	0.64277	37

Table 5.5: Louvain with modularity on weighted graphs.

Clusters	Modularity	NMI	ARI
15	0.06745	0.9386	0.78258

Threshold-based approach

In this experiment we tried different values of parameter $t \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and kept as final solution, the one with the maximum inclusion score. Then we applied Louvain algorithm to the same graph. The results are presented in tables 5.6 and 5.7.

Table 5.6: GNM with threshold-based approach on weighted graphs.

Clusters	Inclusion	NMI	ARI	Thres
5	0.54647	0.07767	0.02474	0.1
7	0.56496	0.76542	0.42404	0.3
13	**0.61602**	0.91188	0.71777	0.5
12	0.6116	0.8873	0.63651	0.7
39	0.59778	0.34584	0.13106	0.9

Table 5.7: Louvain with modularity on weighted graphs.

Clusters	Modularity	NMI	ARI
14	0.06647	0.92034	0.71777

As it can be observed both kNN-based and threshold-based approaches, end up to a satisfactory enough solution, but worse compared to modularity. This is most

likely due to the loss of information after transforming the graph from weighted to unweighted.

Another obvious disadvantage of these two approaches, is that they have a parameter (k or t) which adds one more level of complexity, that is to find the best parameter value, if no prior knowledge exist on the dataset.

5.3 Weighted graphs

To evaluate the generalized inclusion for weighted similarity matrices, we first conducted experiments on graphs of all the categories of synthetic graphs mentioned in the first section, having 500 vertices and 10 clusters. In order to make the graph weighted, we followed the same approach mentioned in the previous section. We generated the weights from two Normal distributions, one with mean 0.8 and std 0.1 for the existing edges of the graph and another one with mean 0.2 and std 0.1 for the non-existing edges. The results of 5 different executions for each graph type, comparing GNM to Louvain algorithm are summarized in table 5.8.

Table 5.8: GNM (inclusion) vs Louvain on weighted synthetic graphs consisting of 500 vertices and 10 clusters.

	GNM (Inclusion)				**Louvain algorithm (Modularity)**			
Graph	**Clusters**	**Inclusion**	**NMI**	**ARI**	**Clusters**	**Modularity**	**NMI**	**ARI**
B&D	10	0.5951	1	1	9.8	0.1234	0.9682	0.9027
B&DD	9.2	0.5678	0.9748	0.9202	9	0.0894	0.9682	0.9027
DL&SS	9.6	0.6284	0.9901	0.9924	5	0.1498	0.8567	0.7824
DS&DD	8.4	0.5722	0.9706	0.9725	6	0.0901	0.9048	0.8736
IS&DD	8.4	0.6173	0.9501	0.9720	4	0.1395	0.7756	0.6349

It can be observed that in all experiments GNM with inclusion presented much better results than Louvain algorithm with modularity.

Another set of experiments we conducted for the evaluation of generalized inclusion was on weighted ring graphs. An unweighted ring graph consists of complete subgraphs which are connected to each other with only one edge forming a ring. An example of a ring graph with 30 vertices divided in 6 fully connected subgraphs is presented in figure 5.1.

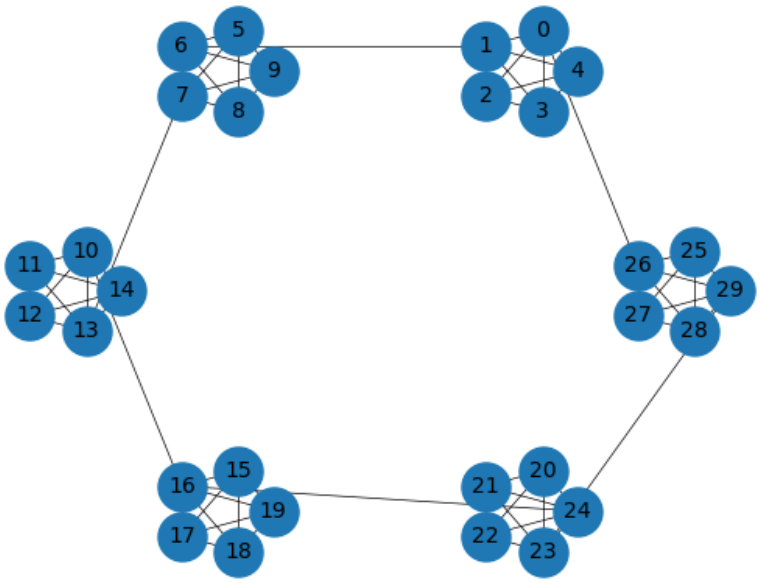


Figure 5.1: Ring graph example.

We defined 5 different categories of ring graphs based on the number of vertices and clusters (table 5.9) and 5 different variations using different Normal distributions for edge weights generation (table 5.10).

Table 5.9: Ring graph categories.

Category	Vertices	Clusters
1	30	10
2	60	15
3	150	30
4	300	50
5	500	10

Table 5.10: Ring graph variations.

	0 - Edges		1 - Edges	
Var.	**Mean**	**Std**	**Mean**	**Std**
1	0	0	1	0
2	0	0	0.8	0.1
3	0.1	0.1	0.8	0.1
4	0.2	0.1	0.8	0.1
5	0.3	0.2	0.7	0.2

The results from 5 different executions of GNM and Louvain algorithm for each combination of the different ring graph categories and variations, are presented below.

Table 5.11: Ring graphs with 30 vertices and 10 clusters.

	GNM (Inclusion)				Louvain algorithm (Modularity)			
Variation	**Clusters**	**Inclusion**	**NMI**	**ARI**	**Clusters**	**Modularity**	**NMI**	**ARI**
1	10	0.8961	1	1	5.6	0.6720	0.8470	0.5613
2	10	0.8906	1	1	5.8	0.6739	0.8550	0.5735
3	10	0.6601	1	1	6	0.2485	0.8588	05801
4	9.4	0.6020	0.9810	0.9213	5.4	0.1325	0.8245	0.5159
5	6.4	0.5754	0.6897	0.3424	3.6	0.0734	0.6101	0.2606

Table 5.12: Ring graphs with 60 vertices and 15 clusters.

	GNM (Inclusion)				Louvain algorithm (Modularity)			
Variation	**Clusters**	**Inclusion**	**NMI**	**ARI**	**Clusters**	**Modularity**	**NMI**	**ARI**
1	15	0.9407	1	1	8	0.7949	0.8644	0.5874
2	15	0.9350	1	1	9	0.7977	0.8858	0.6275
3	15	0.6316	1	1	9.4	0.2131	0.8923	0.6391
4	14.6	0.5803	0.9931	0.9654	7.4	0.1125	0.8370	0.5157
5	9.6	0.5524	0.7196	0.3785	4.6	0.0592	0.5875	0.2265

Table 5.13: Ring graphs with 150 vertices and 30 clusters.

	GNM (Inclusion)				Louvain algorithm (Modularity)			
Variation	**Clusters**	**Inclusion**	**NMI**	**ARI**	**Clusters**	**Modularity**	**NMI**	**ARI**
1	30	0.9614	1	1	16.6	0.8866	0.8998	0.6283
2	30	0.9585	1	1	17.2	0.8883	0.9047	0.6394
3	30	0.5804	1	1	14.4	0.1390	0.8692	0.5364
4	29.8	0.5455	0.9986	0.9918	11	0.0690	0.8185	0.4308
5	10.8	0.5317	0.5578	0.1588	6.6	0.0379	0.4931	0.1344

Table 5.14: Ring graphs with 300 vertices and 50 clusters.

	GNM (Inclusion)				Louvain algorithm (Modularity)			
Variation	**Clusters**	**Inclusion**	**NMI**	**ARI**	**Clusters**	**Modularity**	**NMI**	**ARI**
1	50	0.9727	1	1	28.2	0.9273	0.9163	0.6488
2	50	0.9712	1	1	27.4	0.9289	0.9129	0.6404
3	50	0.5535	1	1	18	0.0953	0.8420	0.4373
4	49	0.5295	0.9964	0.9766	14.6	0.0468	0.8039	0.3609
5	8.4	0.5227	0.4002	0.0633	7.4	0.0272	0.3834	0.0620

Table 5.15: Ring graphs with 500 vertices and 10 clusters.

	GNM (Inclusion)				Louvain algorithm (Modularity)			
Variation	**Clusters**	**Inclusion**	**NMI**	**ARI**	**Clusters**	**Modularity**	**NMI**	**ARI**
1	10	0.9996	1	1	10	0.8992	1	1
2	10	0.9889	1	1	10	0.8992	1	1
3	10	0.7111	1	1	10	0.3460	1	1
4	10	0.6379	1	1	10	0.2023	1	1
5	10	0.5765	1	1	10	0.0984	1	1

The results of the experiments on ring graphs demonstrate modularity's resolution limit problem. In the case where the ring graph consist of 500 vertices in 10 clusters (table 5.15), both inclusion and modularity easily identify the best clusters. But in all other cases, where the number of vertices in clusters is small in relation to the total number of vertices in the graph, modularity fails to identify the best clustering solution. On the other hand GNM provides high quality solutions (NMI = 1 in almost all cases).

5.4 Synthetic 2D datasets

In this section we present another set of experiments, using spectral clustering with inclusion, modularity and silhouette as quality measures on different synthetic datasets consisting of 2-dimensional vectors. For these experiments we used the alpha parameter of inclusion with value $a = 0.75$, in order to focus more on the inter density of

clusters. To construct the similarity matrix, we used an RBF Kernel with $\sigma = 0.5$. To compute silhouette the Euclidean distance has been used.

The 3Wings dataset

The 3Wings dataset consists of 1500 points divided in 3 clusters. The initial dataset along with its ground truth, are presented below.

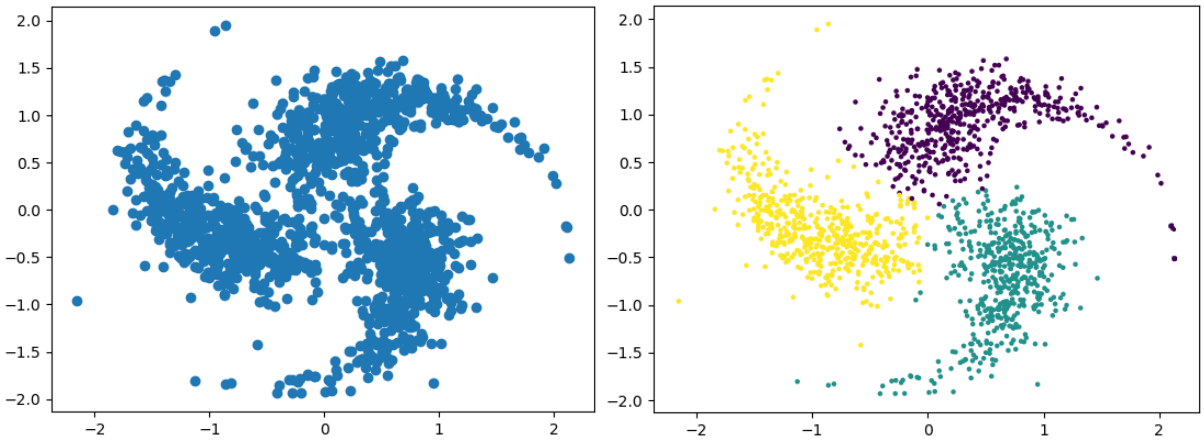


Figure 5.2: The 3Wings dataset and its ground truth.

Next, we present the results of spectral clustering with different number of clusters $k \in \{2, 3, ..., 7\}$, using inclusion, modularity and silhouette for the evaluation of solution.

Table 5.16: Spectral clustering on 3Wings dataset.

NMI	ARI	Clusters	Inclusion	N-Inclusion	Modularity	Silhouette
0.63576	0.52466	2	0.84841	0.84841	0.41195	0.38974
0.88251	0.92547	3	**0.89662**	**0.86478**	**0.62149**	**0.54219**
0.82532	0.87816	4	0.88855	0.76911	0.61962	0.53494
0.77203	0.82191	5	0.87468	0.74436	0.61228	0.48561
0.79906	0.84046	6	0.87735	0.74445	0.61382	0.44878
0.23549	0.05026	7	0.75345	0.71106	0.13407	0.01598

The 5Gaussians dataset

The 5Gaussians dataset consists of 704 points divided in 5 clusters. The initial dataset along with its ground truth, are presented below.

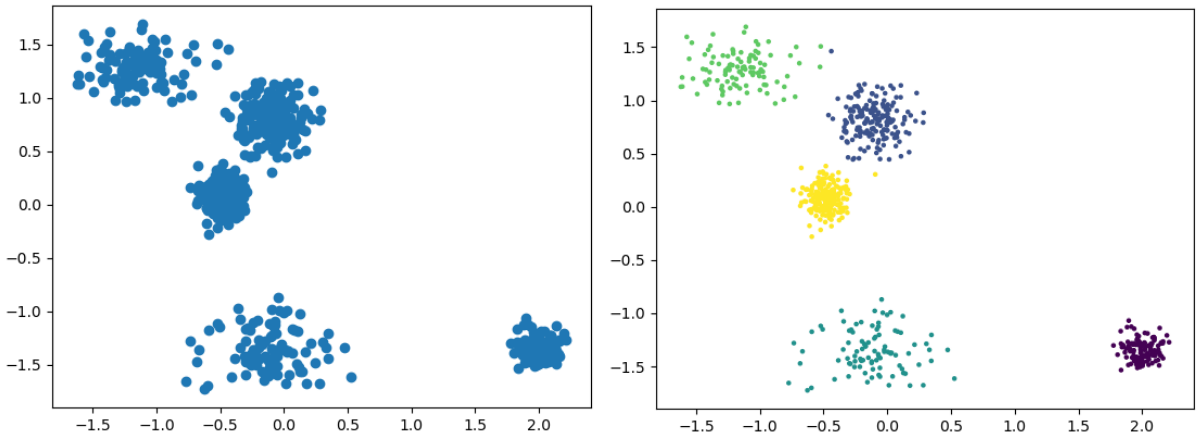


Figure 5.3: The 5Gaussians dataset and its ground truth.

Next, we present the results of spectral clustering with different number of clusters $k \in \{2, 3, ..., 9\}$, using inclusion, modularity and silhouette for the evaluation of each solution as mentioned.

Table 5.17: Spectral clustering on 5Gaussians dataset.

NMI	ARI	Clusters	Inclusion	N-Inclusion	Modularity	Silhouette
0.45886	0.20468	2	0.83865	0.83865	0.24981	0.6659
0.69387	0.40257	3	0.89231	0.85195	0.3351	0.68697
0.8522	0.64572	4	0.92941	0.87518	0.43762	0.68524
0.97286	0.97681	5	**0.93012**	**0.89986**	**0.60794**	**0.7528**
0.81715	0.61895	6	0.88927	0.76481	0.4181	0.62113
0.79521	0.59417	7	0.84851	0.73502	0.3919	0.54409
0.78245	0.58563	8	0.83396	0.72814	0.38386	0.54189
0.90343	0.91285	9	0.83447	0.74749	0.55299	0.65049

The 7Clusters dataset

The 7Clusters dataset consists of 1400 points divided in 7 clusters. The initial dataset along with its ground truth, are presented below.

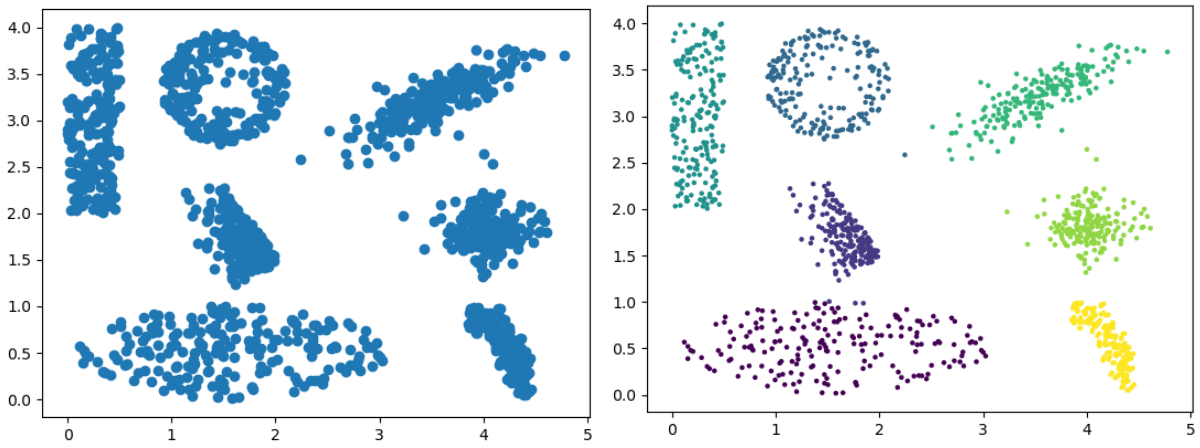


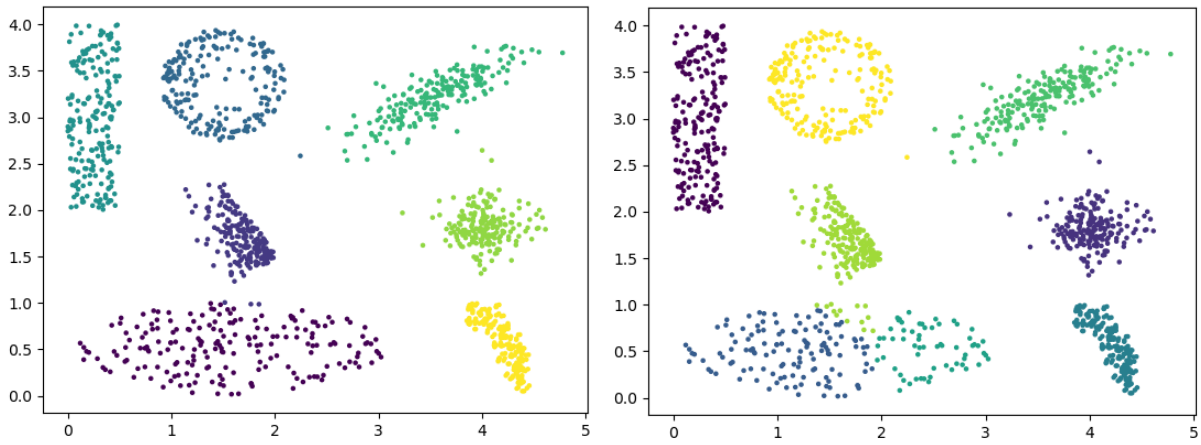
Figure 5.4: The 7Clusters dataset and its ground truth.

Next, we present the results of spectral clustering with different number of clusters $k \in \{2, 3, ..., 11\}$ using inclusion, modularity and silhouette for the evaluation of each solution.

Table 5.18: Spectral clustering on 7Clusters dataset.

NMI	ARI	Clusters	Inclusion	N-Inclusion	Modularity	Silhouette
0.49766	0.26696	2	0.88038	0.88038	0.49874	0.48442
0.63902	0.35355	3	0.9014	0.83776	0.5982	0.46271
0.79996	0.61785	4	0.94177	0.86098	0.70977	0.56075
0.86505	0.71358	5	0.94684	0.87117	0.77503	0.57994
0.91994	0.82813	6	0.94849	0.87854	0.7929	0.5531
0.97504	0.97562	7	**0.94968**	**0.88443**	**0.80519**	0.57846
0.94695	0.93737	8	0.94075	0.83449	0.79846	**0.60051**
0.92395	0.89129	9	0.92601	0.82685	0.78476	0.59817
0.90215	0.84256	10	0.9028	0.81246	0.75784	0.57721
0.90511	0.84414	11	0.89323	0.7917	0.75323	0.56641

In all the experiments we can see that the three quality measures agree to the correct among the solutions obtained from spectral clustering except for the last dataset, where silhouette prefers the solution with the 8 clusters after breaking down in two, the large cluster on the bottom left corner (figure 5.5).



(a) $k = 7$ (b) $k = 8$

Figure 5.5: Spectral clustering solutions on 7Clusters dataset.

5.5 Pointwise inclusion

In this set of experiments, we used the 5Gaussians dataset presented in the previous section and the 4Gaussians, one more dataset consisting again from 2-dimentional vectors generated from four scaled Gaussian distributions, shown in figure 5.6 along with its ground truth.

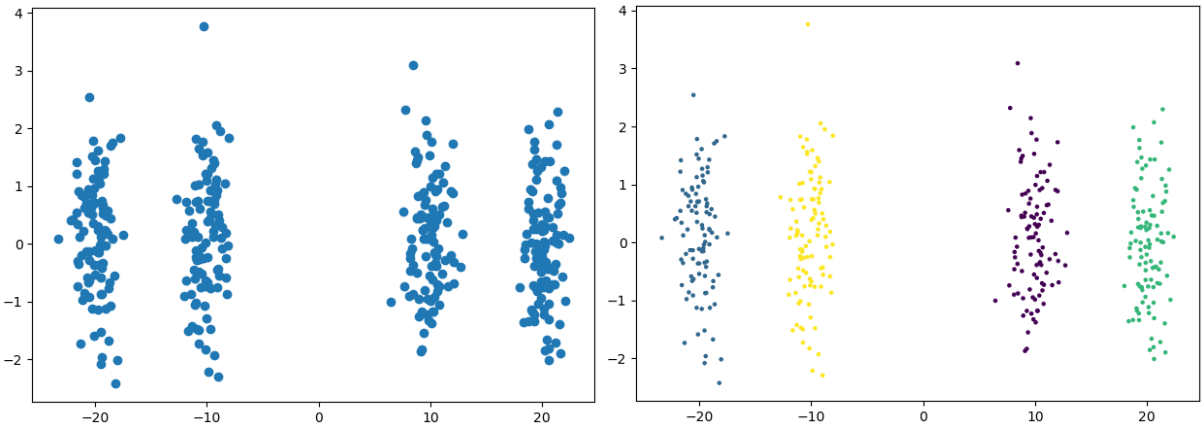


Figure 5.6: The 4Gaussians dataset and its ground truth.

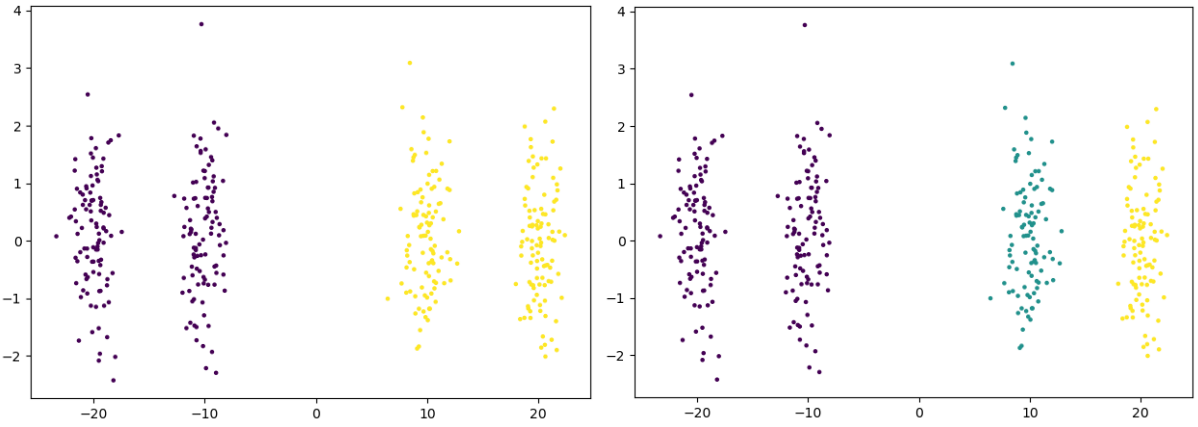
To find the best clustering solution for the data, we applied again spectral clustering for different number of clusters $k \in \{2, 3, ..., 10\}$ and evaluated each solution using inclusion with $a = 0.5$, modularity and silhouette as quality measures. To construct the similarity matrix, we used again an RBF Kernel with $\sigma = 1.5$ for the 4Gaussians

and $\sigma = 0.5$ for the 5Gaussians dataset. For each dataset we also present the histogram of the best object-level inclusion scores. The results of the experiments are presented in tables 5.19 and 5.20.

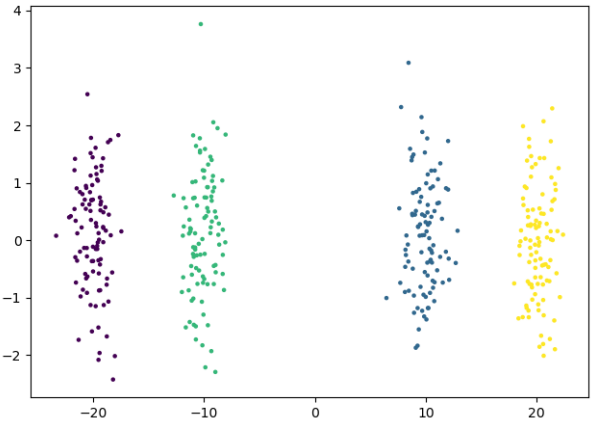
Table 5.19: Spectral clustering on 4Gaussians dataset.

NMI	**ARI**	**Clusters**	**Inclusion**	**N-Inclusion**	**Modularity**	**Silhouette**
0.57716	0.33166	2	0.70626	0.70626	0.36579	0.26226
0.85714	0.71274	3	0.84301	0.74722	0.61592	0.19277
1	1	4	**0.9116**	**0.80564**	**0.7497**	**0.81817**
0.94119	0.91236	5	0.89506	0.71395	0.715	0.71172
0.88913	0.81723	6	0.87158	0.69371	0.66501	0.58581
0.84237	0.71242	7	0.84622	0.67737	0.61119	0.46656
0.80193	0.60375	8	0.82314	0.66013	0.56457	0.34596
0.77918	0.56059	9	0.81024	0.6529	0.53876	0.34989
0.75924	0.52356	10	0.79989	0.65066	0.51848	0.35227

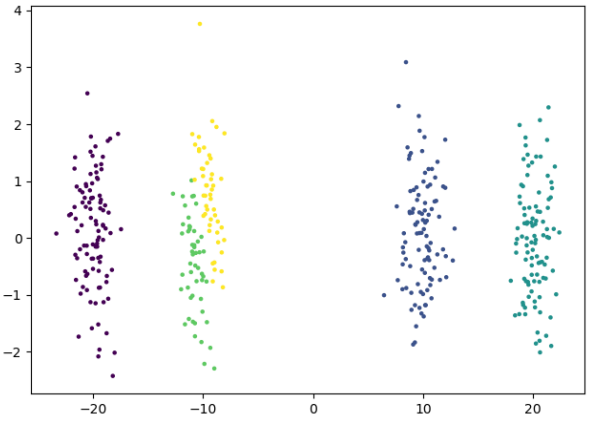
For the experiment on 4Gasussian dataset, we also present the visualization of spectral clustering results for each k.



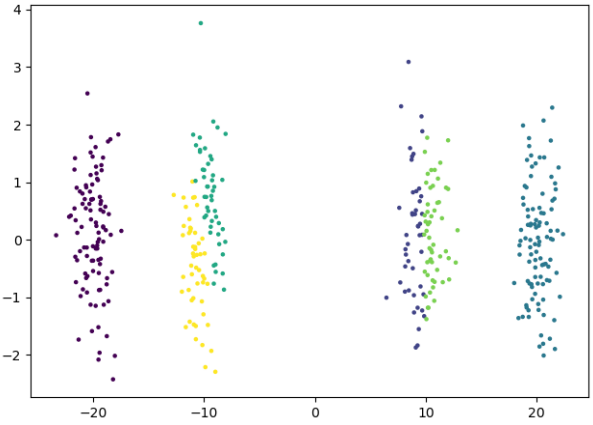
(a) $k = 2$ (b) $k = 3$



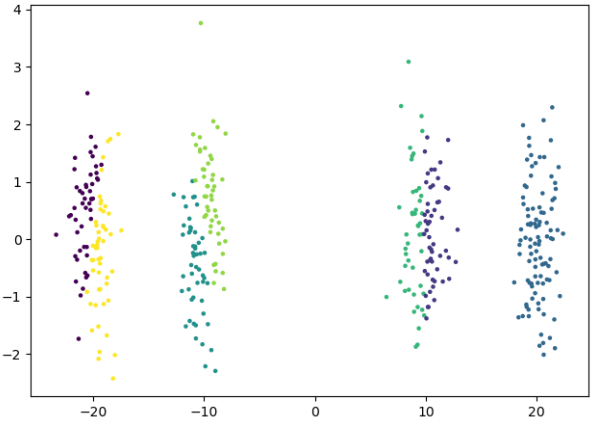
(c) $k = 4$



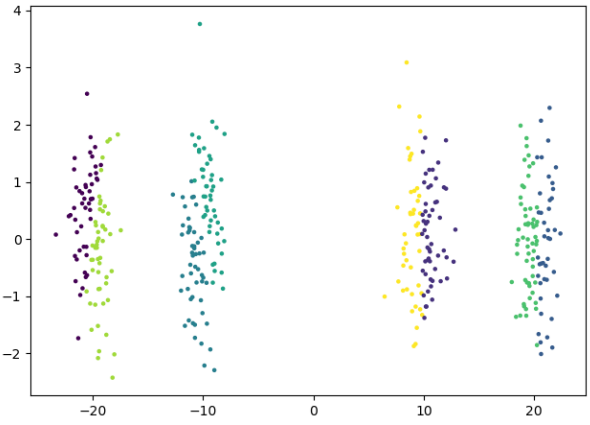
(d) $k = 5$



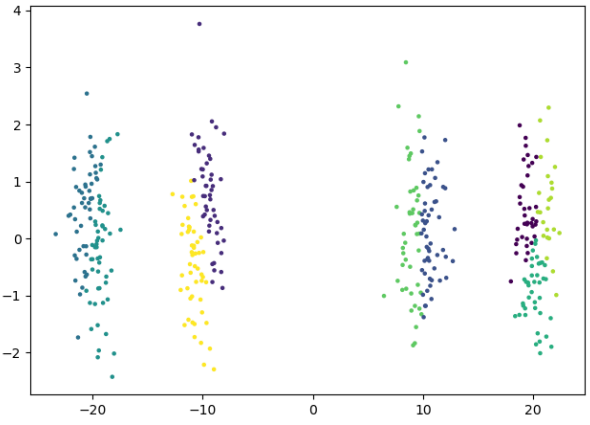
(e) $k = 6$



(f) $k = 7$



(g) $k = 8$



(h) $k = 9$

Figure 5.7: Spectral clustering results on 4Gaussians dataset.

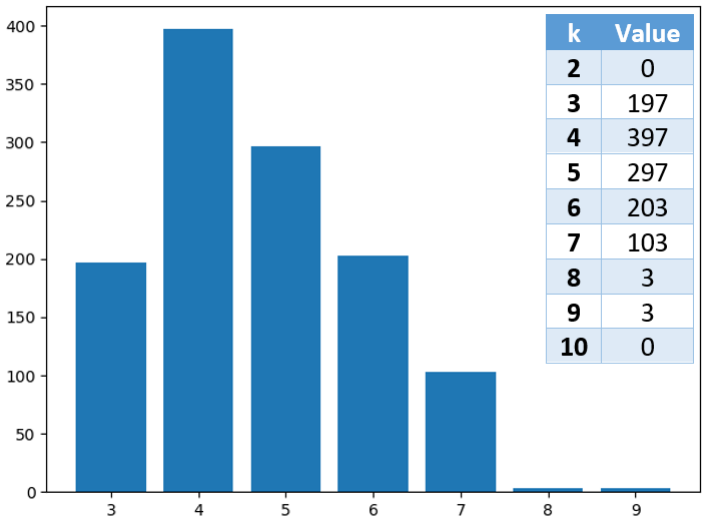


Figure 5.8: Pointwise inclusion histogram for 4Gaussians dataset.

As we mentioned, the histogram of the best object-level inclusion scores can be used to give us additional information for the clustering solution. In the case of 2 clusters none of the points are presented their maximum inclusion score. In case of 3 clusters we can see that the 2 out of 4 clusters are separated correctly and therefore we observe in the histogram that almost half of the data have their maximum inclusion score. In case of 4 clusters almost all the data are well placed. For number of clusters $k \in \{5, 6, 7, 8\}$ a cluster is split in half each time meaning that objects in that cluster get lower inclusion score. For this reason the data in this cluster are removed from the corresponding histogram bin.

Table 5.20: Spectral clustering on 5Gaussians dataset.

NMI	**ARI**	**Clusters**	**Inclusion**	**N-Inclusion**	**Modularity**	**Silhouette**
0.45886	0.20468	2	0.6773	0.6773	0.24981	0.6659
0.69387	0.40257	3	0.78565	0.70419	0.3351	0.68697
0.8522	0.64572	4	0.87394	0.76055	0.43762	0.68524
0.97286	0.97681	5	**0.93305**	**0.8607**	**0.60794**	**0.7528**
0.81715	0.61895	6	0.84915	0.5831	0.4181	0.62113
0.79521	0.59417	7	0.82528	0.58294	0.3919	0.54409
0.78245	0.58563	8	0.81604	0.58006	0.38386	0.54189
0.90343	0.91285	9	0.87409	0.6275	0.55299	0.65049
0.88798	0.90126	10	0.86264	0.62535	0.54106	0.6414

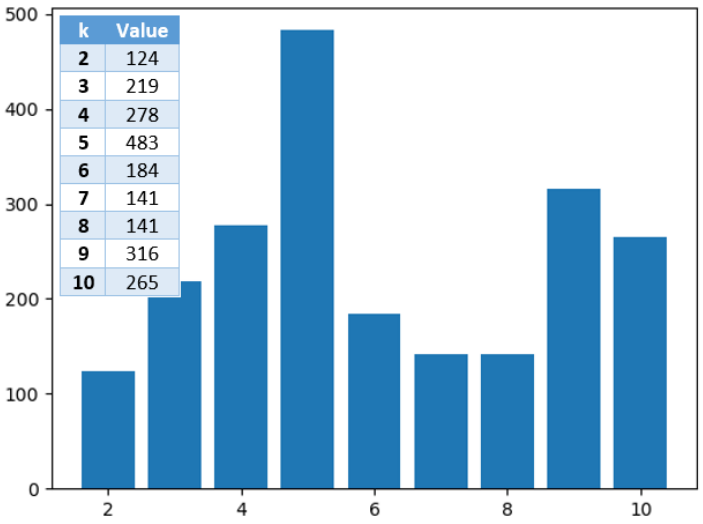


Figure 5.9: Pointwise inclusion histogram for 5Gaussians dataset.

5.6 Inclusion on real datasets

In this section we consider three real-world datasets: Pendigits, Optdigits and Iris. Pendigits[2] train set consists of 7494 digits of 10 classes produced from 44 writers using a pressure sensitive tablet. Each digit has 16 attributes and its label.

Optdigits[3] train set consists of 3823 hand-written digits of 10 classes and each digit has 64 attributes and its label. For both datasets, to produce the similarity matrix, we used an RBF Kernel with $\sigma = 0.7$. We applied spectral clustering for various subsets of digits ($\{2,3\}, \{2,3,6\}, \{0,2,3,6\}, \{0,2,4,6,8\}, \{1,3,5,7,9\}$) and determined the best solution using inclusion, modularity and silhouette. The results of the experiments on Pendigits and Optdigits datasets are presented in tables 5.21 - 5.28.

Table 5.21: Spectral clustering on Pendigits subset {2, 3}.

NMI	ARI	Clusters	Inclusion	N-Inclusion	Modularity	Silhouette
1	1	2	**0.81444**	**0.81444**	**0.48201**	**0.59598**
0.81229	0.79929	3	0.80743	0.64553	0.45818	0.47272
0.69638	0.58648	4	0.76808	0.62059	0.39502	0.33518

[2]https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits
[3]https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

Table 5.22: Spectral clustering on Pendigits subset {2, 3, 6}.

NMI	ARI	Clusters	Inclusion	N-Inclusion	Modularity	Silhouette
0.74252	0.5908	2	0.75356	0.75356	0.4425	0.40546
1	1	3	**0.86787**	**0.8012**	**0.62459**	**0.55881**
0.90303	0.87678	4	0.84135	0.691	0.57488	0.48362
0.82941	0.76212	5	0.82477	0.64283	0.54485	0.41905

Table 5.23: Spectral clustering on Pendigits subset {0, 2, 3, 6}.

NMI	ARI	Clusters	Inclusion	N-Inclusion	Modularity	Silhouette
0.48851	0.28118	2	0.68819	0.68819	0.28693	0.38716
0.8148	0.69844	3	0.83053	0.74359	0.62116	0.41473
0.91949	0.90816	4	**0.88525**	**0.7849**	**0.69435**	**0.52609**
0.90213	0.87776	5	0.87545	0.70466	0.66988	0.49907
0.90708	0.87961	6	0.869	0.62211	0.65852	0.48879

Table 5.24: Spectral clustering on Pendigits subset {0, 2, 4, 6, 8}.

NMI	ARI	Clusters	Inclusion	N-Inclusion	Modularity	Silhouette
0.384	0.22456	2	0.69277	0.69277	0.403	0.26311
0.59411	0.4775	3	0.79913	**0.72015**	0.56811	0.30085
0.71062	0.61062	4	0.83313	0.71891	0.63921	0.34925
0.79767	0.74548	5	0.85531	0.71041	**0.66094**	0.40603
0.83497	0.79325	6	**0.85834**	0.68005	0.65911	**0.41883**
0.8287	0.77326	7	0.84907	0.68173	0.63689	0.41363

Table 5.25: Spectral clustering on Optdigits subset {2, 3}.

NMI	ARI	Clusters	Inclusion	N-Inclusion	Modularity	Silhouette
1	1	2	0.74866	**0.74866**	0.49649	**0.23381**
0.79206	0.75155	3	**0.77875**	0.67555	**0.54739**	0.20273
0.3581	0.29041	4	0.66522	0.50051	0.33785	0.09842

Table 5.26: Spectral clustering on Optdigits subset {2, 3, 6}.

NMI	ARI	Clusters	Inclusion	N-Inclusion	Modularity	Silhouette
0.73148	0.56625	2	0.7212	0.7212	0.49893	**0.27316**
1	1	3	0.83223	**0.74839**	**0.63422**	0.26981
0.90058	0.86867	4	**0.83885**	0.67311	0.62542	0.24642
0.7151	0.61061	5	0.77694	0.50162	0.47169	0.16525

Table 5.27: Spectral clustering on Optdigits subset {0, 2, 3, 6}.

NMI	ARI	Clusters	Inclusion	N-Inclusion	Modularity	Silhouette
0.66666	0.49951	2	0.75034	**0.75034**	0.38283	0.23109
0.85167	0.70729	3	0.81128	0.70724	0.62122	0.25885
0.99671	0.99827	4	**0.87389**	0.74846	**0.65402**	**0.28349**
0.93228	0.90857	5	0.87349	0.67165	0.63996	0.26968
0.78988	0.62767	6	0.81228	0.50389	0.40154	0.19313

Table 5.28: Spectral clustering on Optdigits subset {0, 2, 4, 6, 8}.

NMI	ARI	Clusters	Inclusion	N-Inclusion	Modularity	Silhouette
0.46306	0.20648	2	0.65829	0.65829	0.49947	0.11062
0.76229	0.60336	3	0.81749	0.70023	0.42624	0.18052
0.8852	0.77246	4	0.8572	0.71432	0.65712	0.21729
0.9811	0.98815	5	0.89523	**0.74583**	0.66811	0.23768
0.97057	0.97789	6	**0.89614**	0.52473	**0.66815**	**0.23919**
0.82048	0.68733	7	0.85499	0.52664	0.42783	0.17243

We also applied k-means to the same subsets of digits, but results were similar to the ones of spectral clustering.

For most of the cases, all compared criteria show good results. There are cases though, where some of them deviate from the correct solution. This deviation may be due to the fact that these datasets consists of handwritten digits. Depending on the handwriting, there may be different digits similar to each other in which case a criterion may prefer to combine them into one cluster. On the other hand, the same

digit can be written in different ways so it may be preferable from a criterion, to split the cluster of that digit into two or more clusters.

Finally, Iris[4] is a dataset of 150 objects. Each object consists of 4 attributes and its label, which refers to a type of iris plant. There are 3 different types of plants: Iris Setosa, Iris Versicolour and Iris Virginica. To produce the similarity matrix, we used an RBF Kernel with $\sigma = 0.5$. The results of the experiments are presented in table 5.29.

Table 5.29: Spectral clustering on Iris dataset.

NMI	**ARI**	**Clusters**	**Inclusion**	**N-Inclusion**	**Modularity**	**Silhouette**
0.6994	0.55837	2	**0.76274**	**0.76274**	**0.16193**	**0.68579**
0.77715	0.70736	3	0.71908	0.68669	0.13348	0.55173
0.60559	0.45306	4	0.67977	0.63269	0.09495	0.37892
0.72092	0.64031	5	0.64289	0.6233	0.08031	0.32846

Summarizing the experimental results, inclusion criterion presents in many cases better results than modularity. Compared to silhouette criterion, inclusion has in general equally good results. In regards to comparison between inclusion and nearest inclusion criterion, nearest inclusion presents good results too, but in difficult datasets like Pendigits and Optdigits, it seems to prefer solutions with small number of clusters, deviating from the real solution.

[4]https://archive.ics.uci.edu/ml/datasets/iris

Chapter 6

Conclusions and Future work

In this thesis we studied data clustering focusing on inclusion, a recently proposed quality measure to evaluate a clustering solution, along with Greedy Node Movement, an algorithm for optimizing this criterion. The inclusion measure can be used to solve clustering problems without a prior knowledge of the number of clusters.

The goal of this thesis is the generalization of inclusion quality measure to general similarity matrices, since until now, it was defined only for binary similarity matrices. To achieve that, we considered three different approaches. Two of them rely on binary inclusion but first, they convert the similarity matrix to binary. The third approach and the most effective relies on extending the inclusion definition to accommodate general similarity values. We propose two definitions of inclusion.

The experiments we conducted showed that the first two approaches that rely on binary inclusion provide good results, but inferior to other criteria like modularity and silhouette, due to the loss of information after converting the similarity matrix to binary.

However, the generalization of inclusion itself had very good experimental results and in many cases much better than the previously mentioned quality measures.

Even though inclusion presented very good results, there are several issues that need further elaboration. An important one is to assess the performance of inclusion on more datasets, especially real-world ones. The two versions of inclusion, the role of parameter a and the comparative performance with respect to silhouette should be investigated in more detail.

Another issue that could be studied concerns the development of greedy heuristic of lower computational complexity compared to Greedy Node Movement, that would allow the direct optimization of inclusion criterion to partition large datasets.

Bibliography

[1] N. Koufos, "Community detection in undirected graphs using a new quality measure," MSc Thesis, Department of Computer Science And Engineering, University of Ioannina, Jul. 2017.

[2] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining, (First Edition)*. USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[3] A. Pothen, *Graph Partitioning Algorithms with Applications to Scientific Computing*. Dordrecht: Springer Netherlands, 1997, pp. 323–368.

[4] C. Augeri and H. Ali, "New graph-based algorithms for partitioning vlsi circuits," vol. 4, 2004, pp. 521–524.

[5] B. Hendrickson and T. G. Kolda, "Graph partitioning models for parallel computing," *Parallel Computing*, vol. 26, no. 12, pp. 1519 – 1534, 2000.

[6] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell System Technical Journal*, vol. 49, no. 2, pp. 291–307, 1970.

[7] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *19th Design Automation Conference*, 1982, pp. 175–181.

[8] Z. Lu, J. Wahlström, and A. Nehorai, "Community detection in complex networks via clique conductance," *Scientific Reports*, vol. 8, no. 1, p. 5982, 2018.

[9] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, 1967, pp. 281–297.

[10] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Adv. Neural Inf. Process. Syst*, vol. 14, pp. 849—856, 2002.

[11] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53 – 65, 1987.

[12] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, p. 026113, 2004.

[13] S. Fortunato and M. Barthélemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences*, vol. 104, no. 1, pp. 36–41, 2007.

[14] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity - np-completeness and beyond," 2006.

[15] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008.

[16] N. Koufos and A. Likas, "The inclusion measure for community evaluation and detection in unweighted networks," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2018, pp. 1053–1056.

[17] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. USA: Wiley-Interscience, 2006.

[18] A. Strehl and J. Ghosh, "Cluster ensembles — a knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, p. 583–617, 2003.

[19] C. Guyeux, S. Chrétien, G. Bou Tayeh, J. Demerjian, and J. Bahi, "Introducing and comparing recent clustering methods for massive data management in the internet of things," *Journal of Sensor and Actuator Networks*, vol. 8, no. 4, 2019.

Short Biography

Nikolaos Kornelakis was born in Athens, Greece in 1994. In 2012, he enrolled in the Department of Computer Engineering, School of Applied Technology, Technological Educational Institute of Epirus and received the BSc degree in 2017. In 2018, he enrolled in the Department of Computer Science & Engineering, University of Ioannina as a MSc student. He has also been working as a back-end software developer since 2019. His main interests are in the areas of data mining and software development.